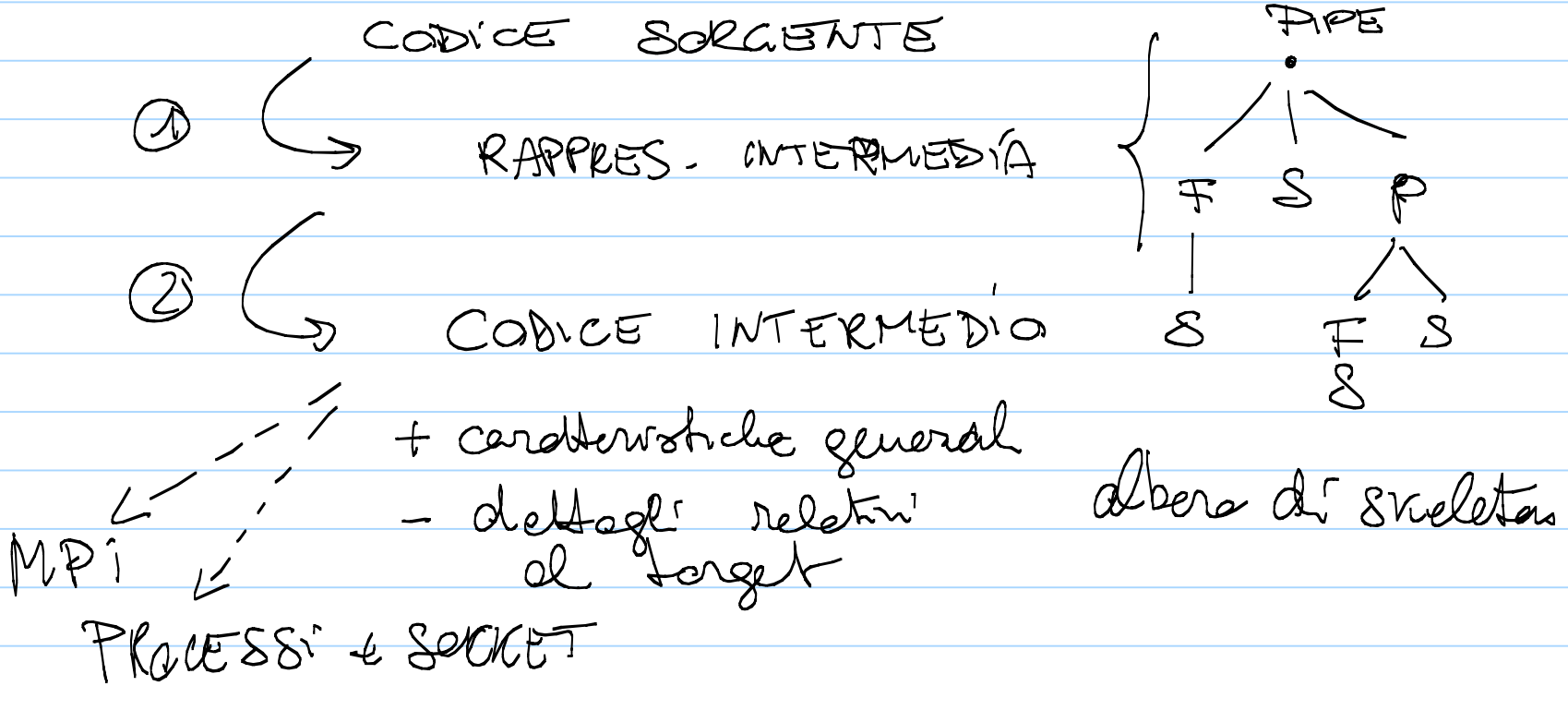
Implementazione di skeleton

Note Title 11/03/2009

Template vs MDF macro data flow



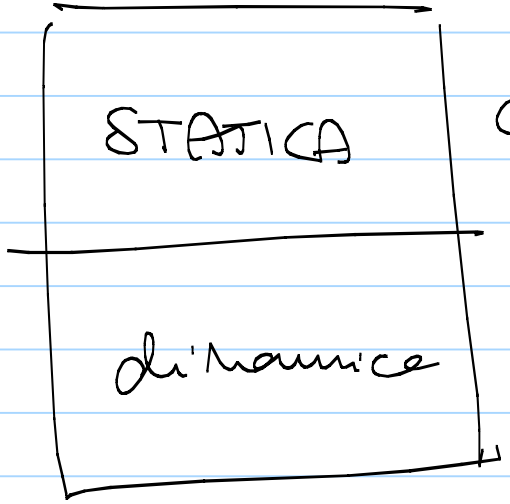
albero di skeleton

③ Compilazione nel "target" ← (macchina hw + lib)

④ RUN = • + RTS

↳ tutte le librerie necessarie e "predefinite"

implementazione

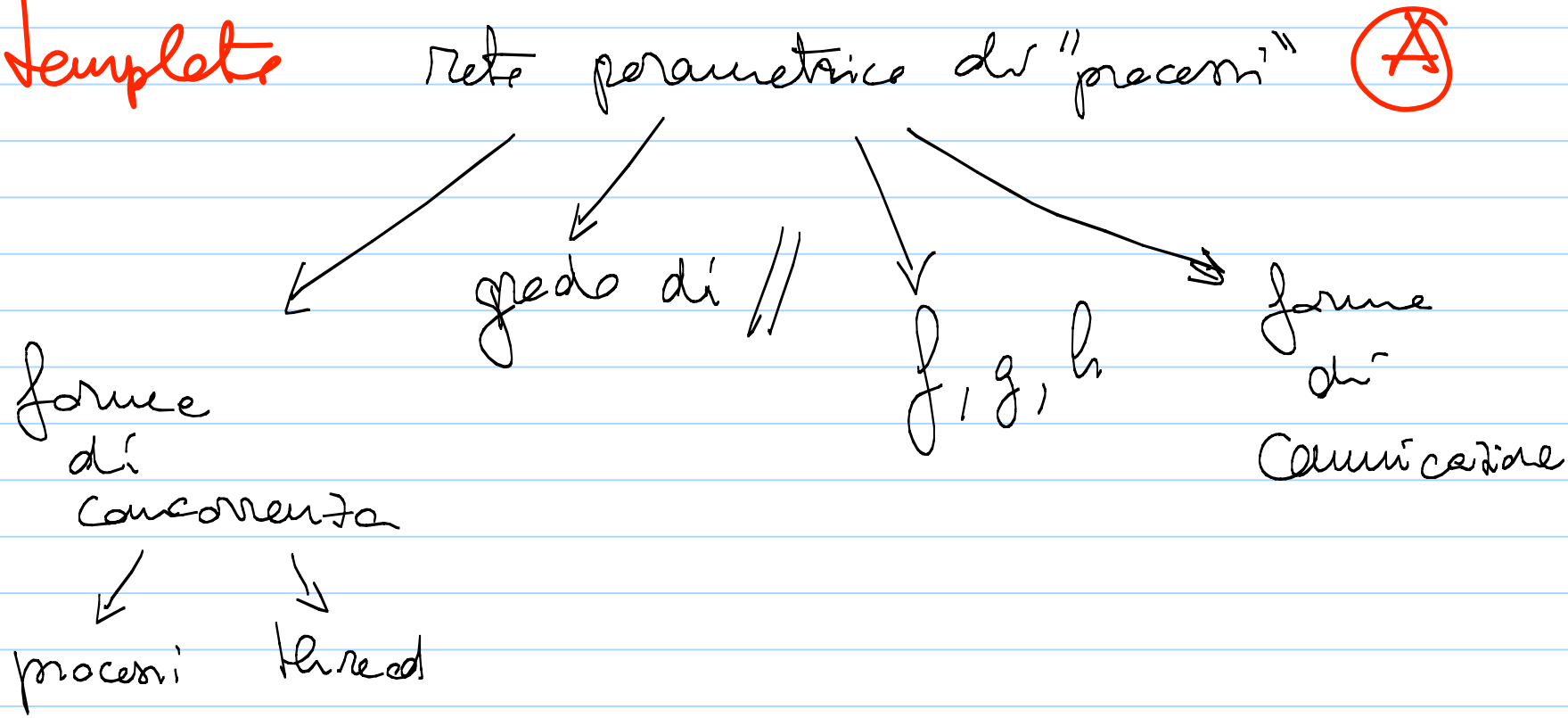


compilatori

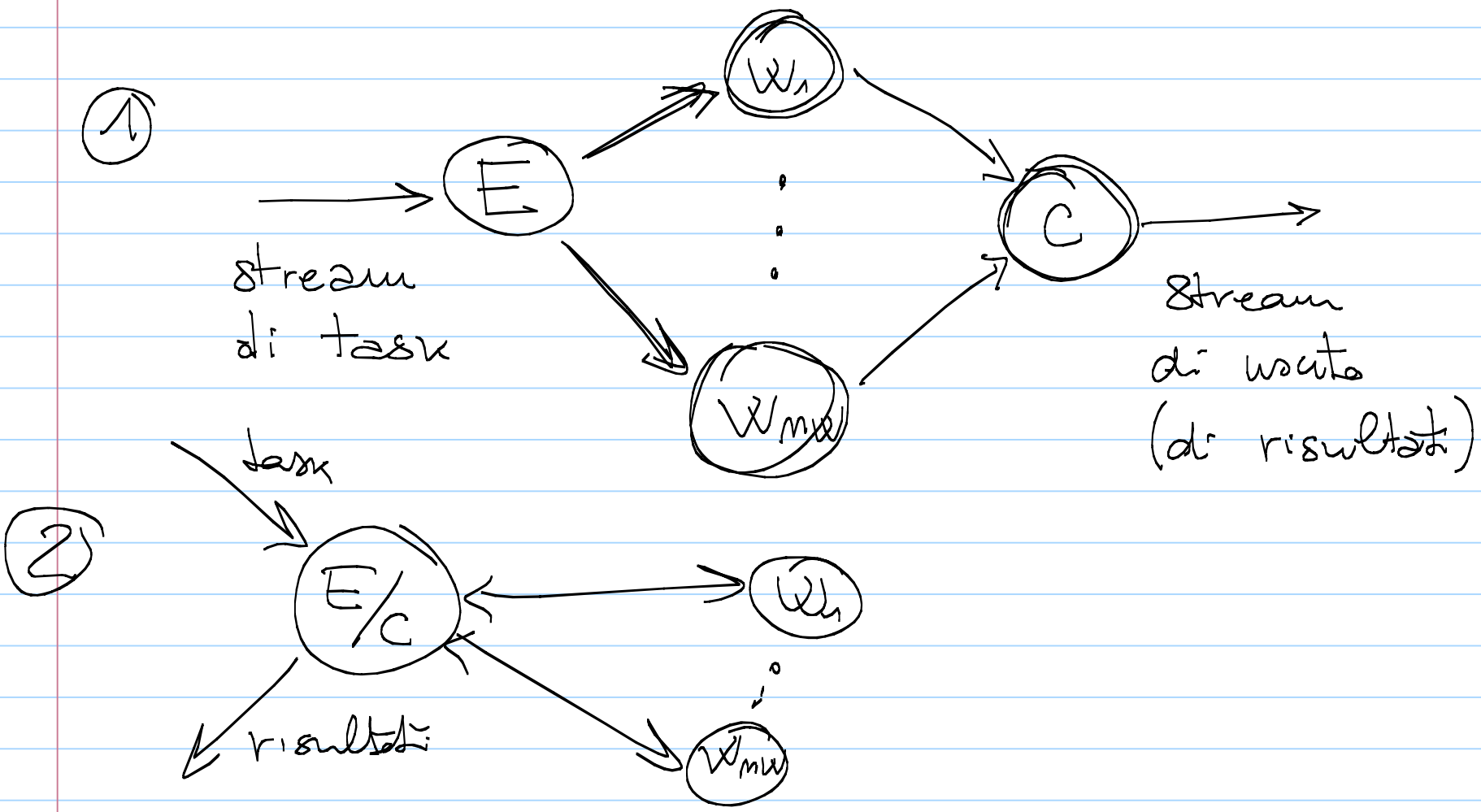
RTS

Implementazione a "template"

template rete parametrica di "processi"

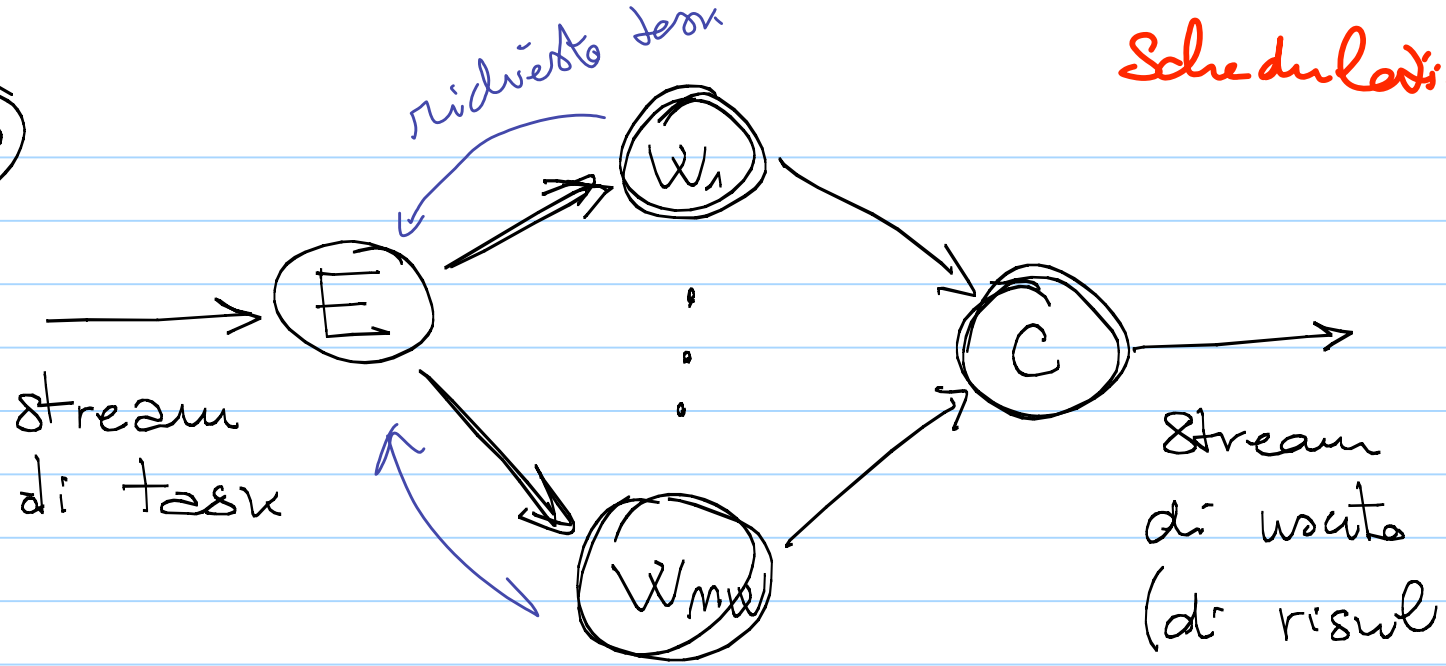


template x task farm



Schedulazione

3



template modello della prestazione

↓

permettere scelta fra template alternativi

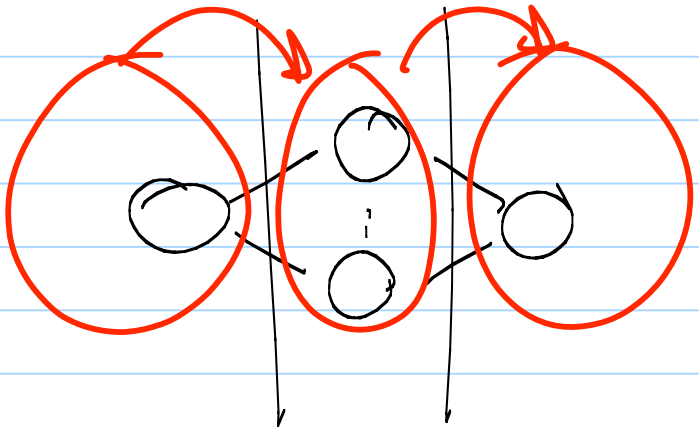
→ abbastanza accurato

↑ parametri { latenza comun.
CPU
μ / DISK

→ "veloce" da calcolare

Esempi di modelli di performance

farm $T_S \approx \frac{T_{S_W}}{N_W}$



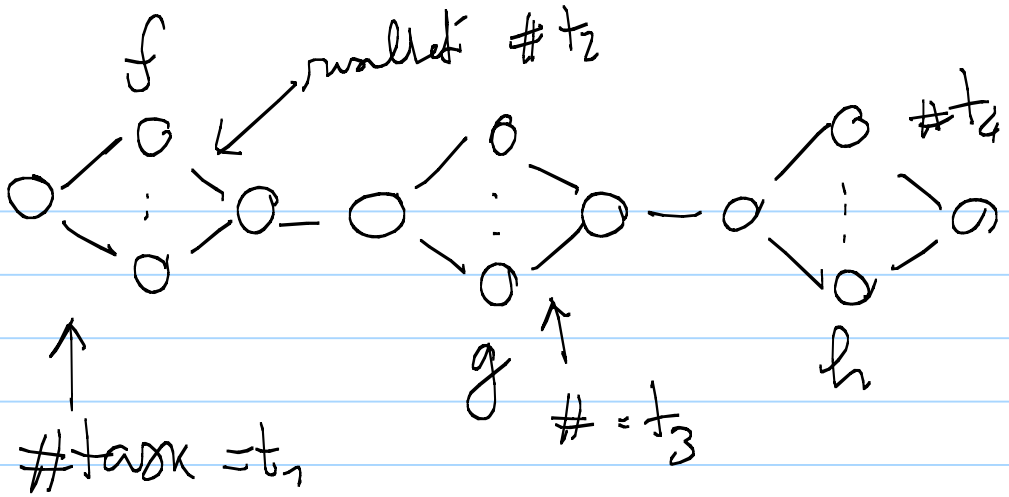
$T_S = \max\{T_{S_1}, T_{S_2}, T_{S_3}\}$

$$t_{comm}(k) = t_0 + k t_1$$

$$T_{S_N} = T_f + T_{comm}(\#task)$$

$$T_S = \frac{\left(T_f + T_{comm}(\#task)\right)}{N_W}$$

Caratterizza meglio la computazione
(e quindi dà un miglior
modello analitico)



$f : \alpha \to \beta$

$g : \beta \to \gamma$

$h : \gamma \to \delta$

$\# \delta = t_4$

Per il modello di performance:

Compromesso

precisione

presenta

piccoli errori nei parametri si ripercuotono nel modello

efficienza del calcolo

rende

sono "veri cicli di clock" consumati x calcolare il modello

90 - 2000 (Aldwinckle e Cappola)

a) modelli molto precisi ($t_{cam} = t_0 + k t_n$)

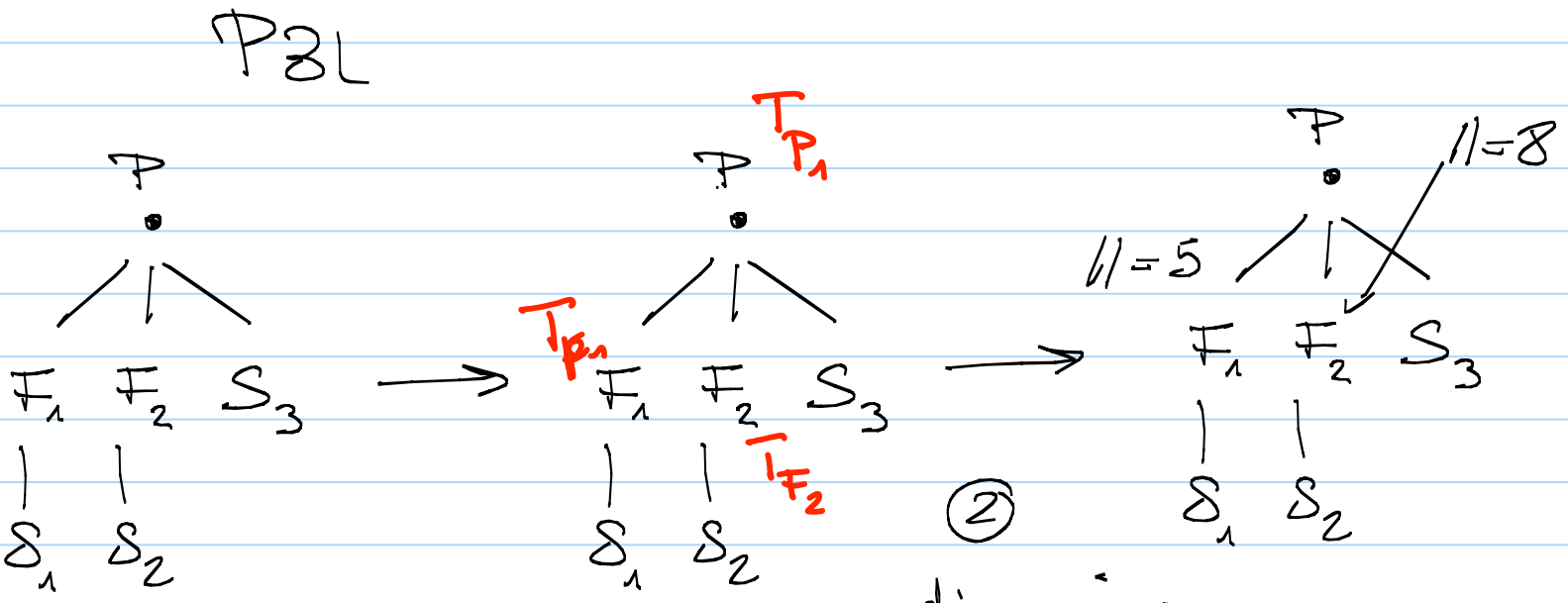
$t_f = \ldots$

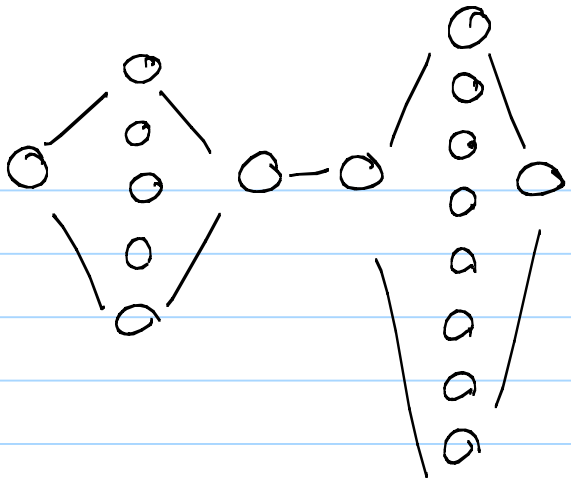
↳ difficile valutare i confronti

xché abbiamo sistemi con poche equazioni e tanto variabili

b) modelli — precisi danno invece risultati comparabili

Esempio utilizzo modelli





17 Risorse necessarie

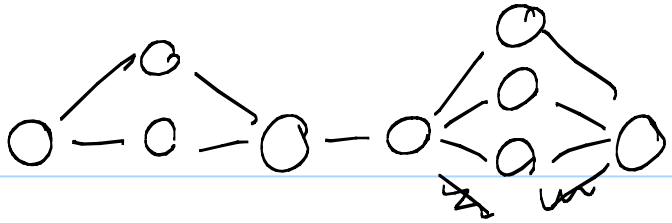
Mia. Ho 8 PE

2ª fase di "RIDUZIONE"

Iterativo:

a) tolgo il minimo numero di risorse che abbia senso

b) fino a che #risorse ≤ #PE



codice intermedio

STOP

MODELLI di PERF.

① → dimensioni massime

② → ridurre → errori

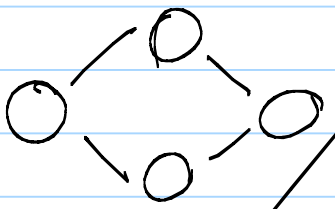
di

software

Metodi di dimensionamento
à la Ricerca Operativa davano risultati migliori

Template

mio schema di implementazione sul target



+ schemi 1 ∀ target

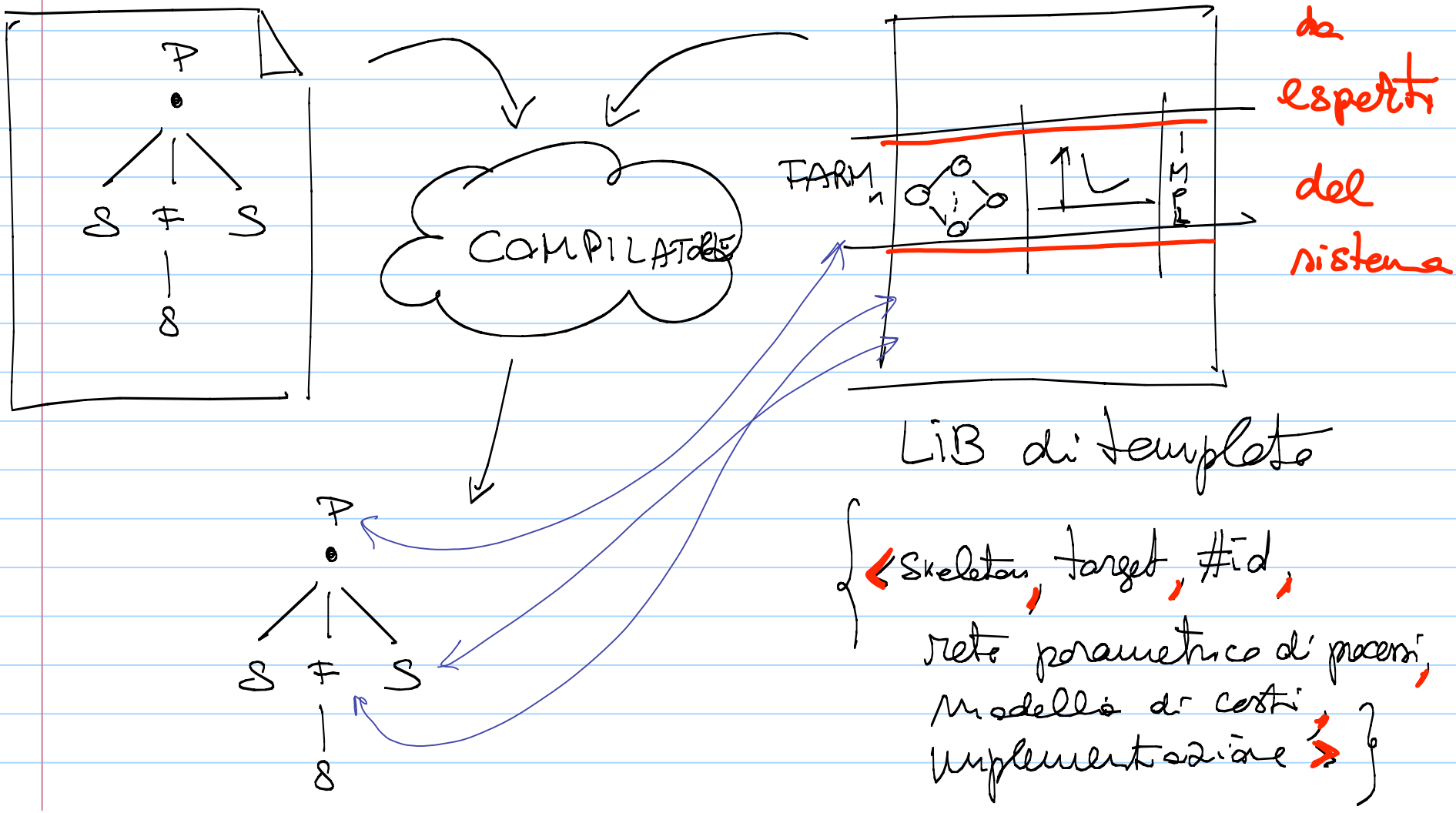
SMP

- thread
  - → caso di memoria shared
  - Synch barriere

MIMD (ow now)

- processi
  - → TCP/IP socket
  - synch protocolli distribuiti

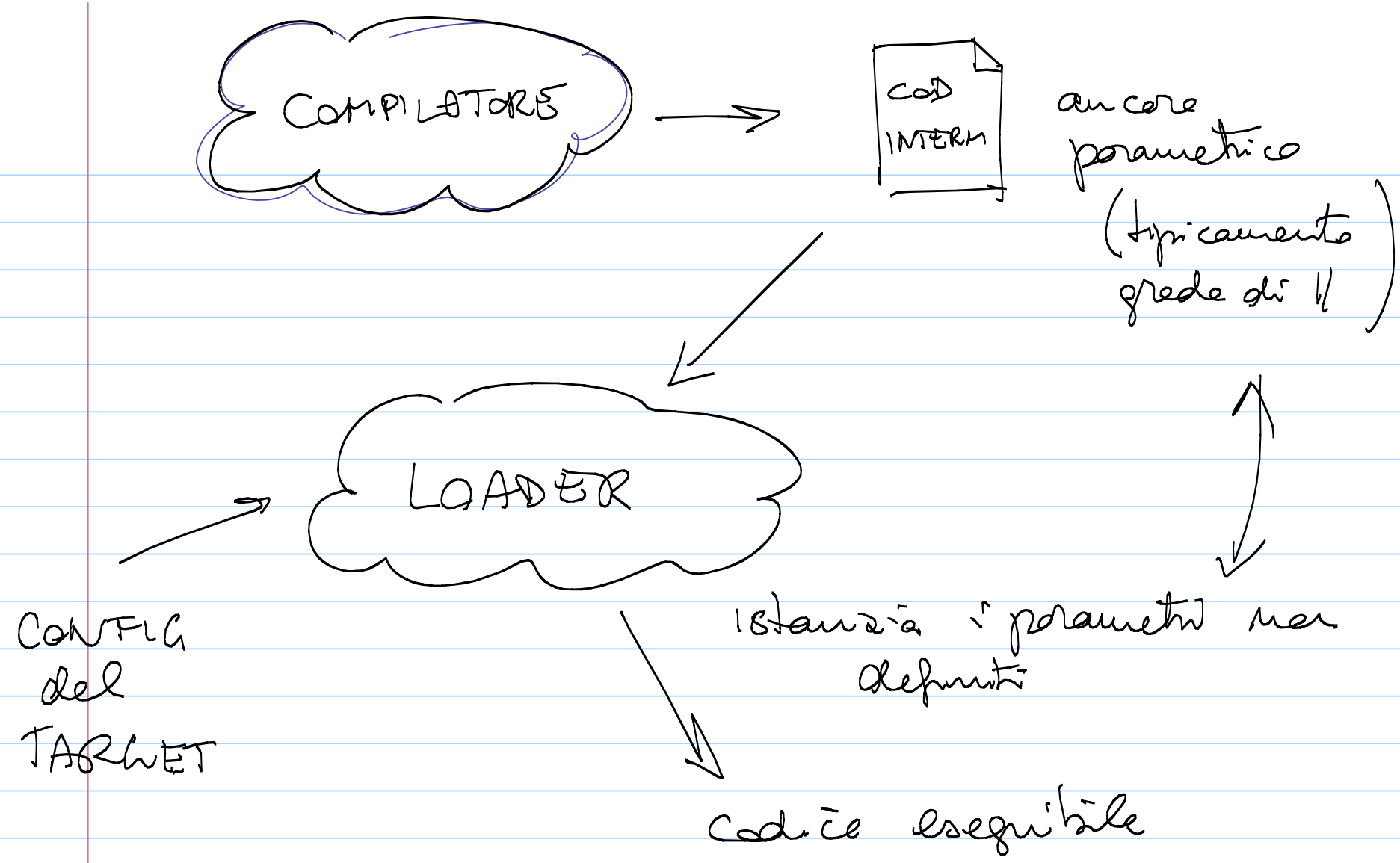
Implementazione a template



SCRITTA da esperti del sistema

LIB di template

{ <Skeleton, target, #id,
rete parametrica di processi,
modello di costi,
implementazione> }


COMPILATORE
COD INTERM
ancora parametrico
(tipicamente grado di ||
LOADER
CONFIG del TARGET
istanzia i parametri non definiti
codice eseguibile

1° modello di implementazione x skeleton

+ diffuso !

→ Muesli (Kuchen C++ + MPI)

eskel (Cole C + MPI)

ASSIST (...)

SkeTo (C+MPI, Kunimori Tokio)

Modello MACRO DATA FLOW

Data flow "normale"

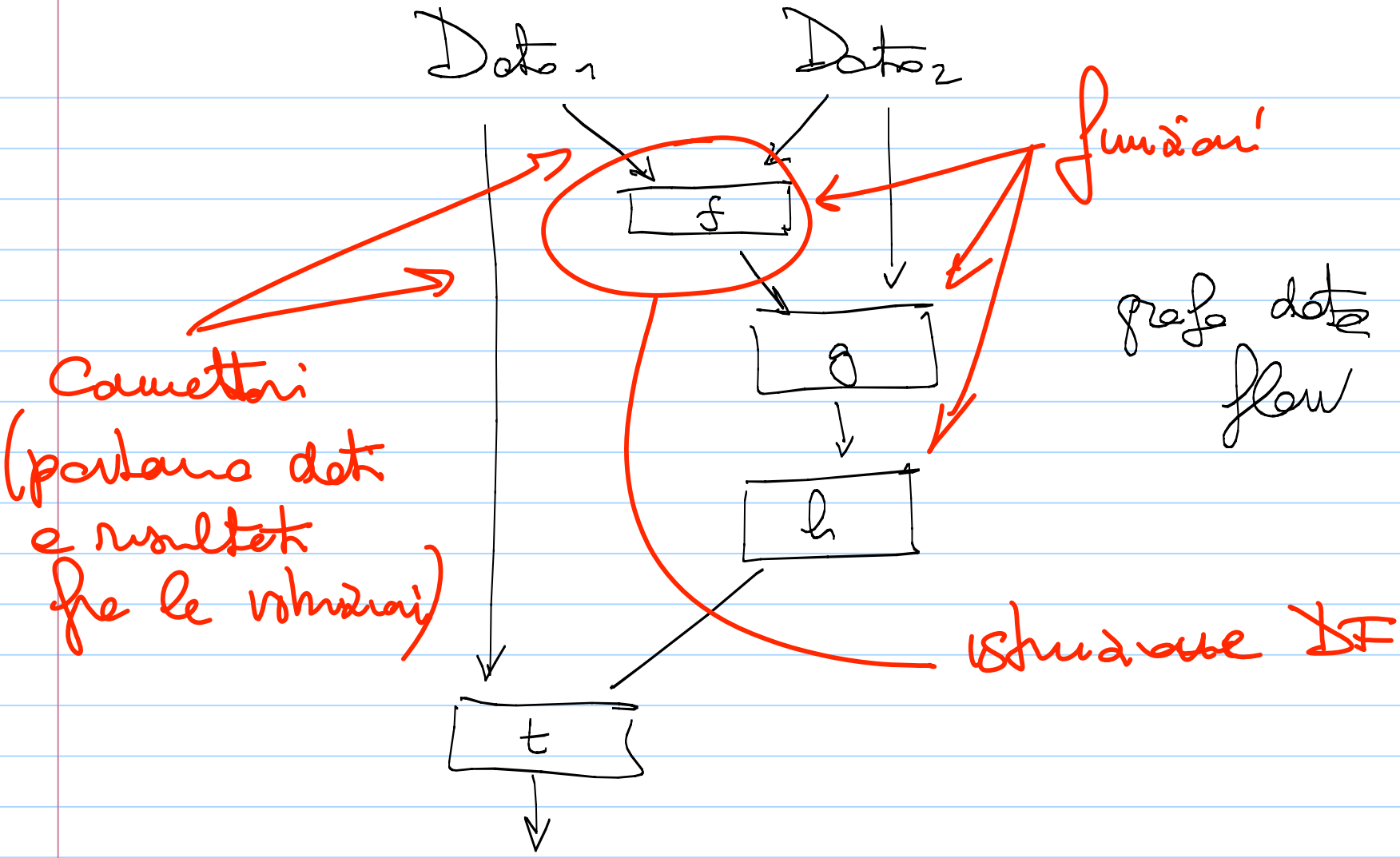
VS Control flow

dicono cosa specifico la mia computazione

Control flow $S_1 ; S_2 ; S_3 \dots$

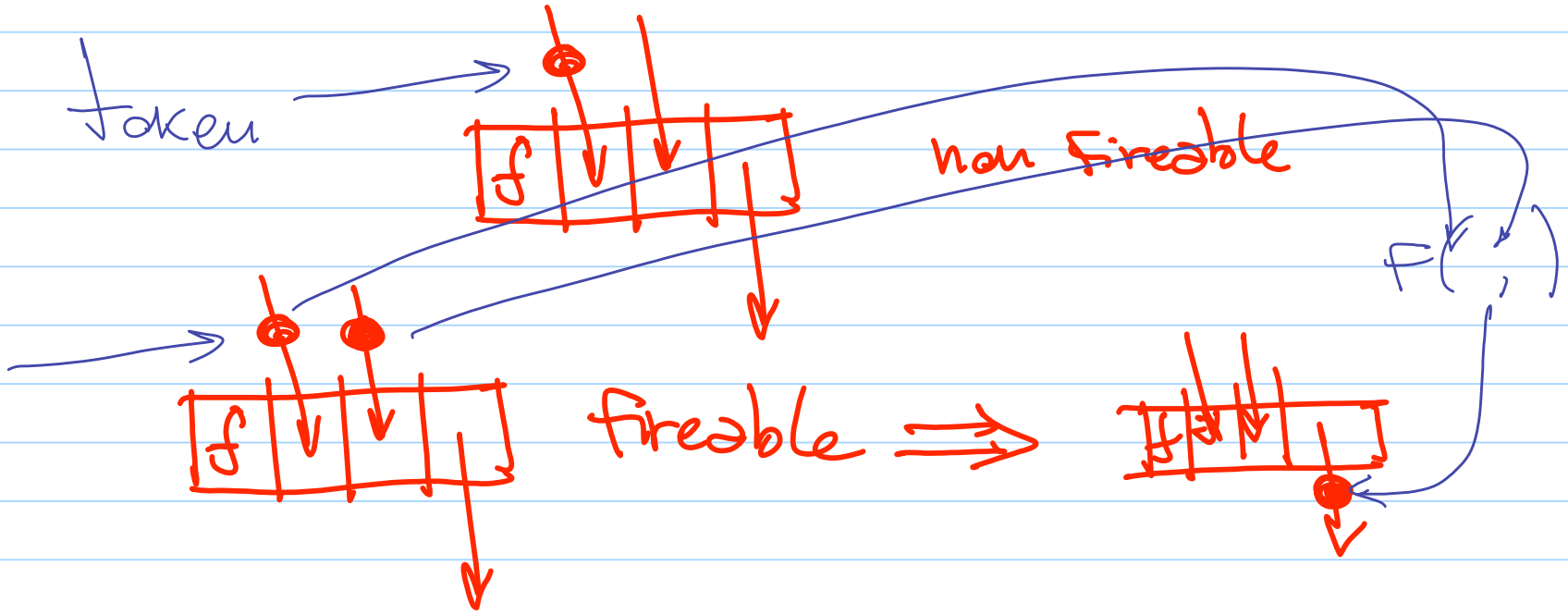
prima S_1 poi S_2 ... $\Rightarrow$ computazione

Data flow dipendenze sui dati


Dato 1
Dato 2
f
g
h
t
funzioni
grafo data flow
connettori
(portano dati
e risultati
fra le istruzioni)
istruzione DF

Firing rule :

posso calcolare l'istruzione che abbia tutti i dati in input



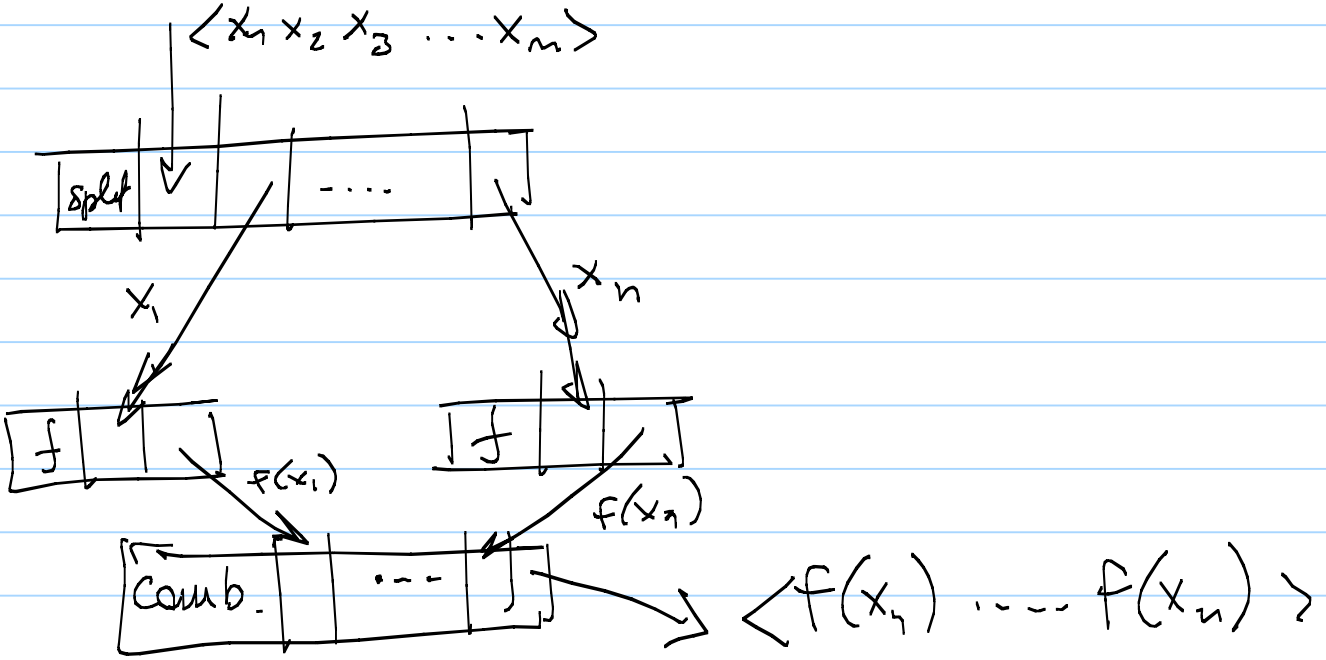
Map data flow

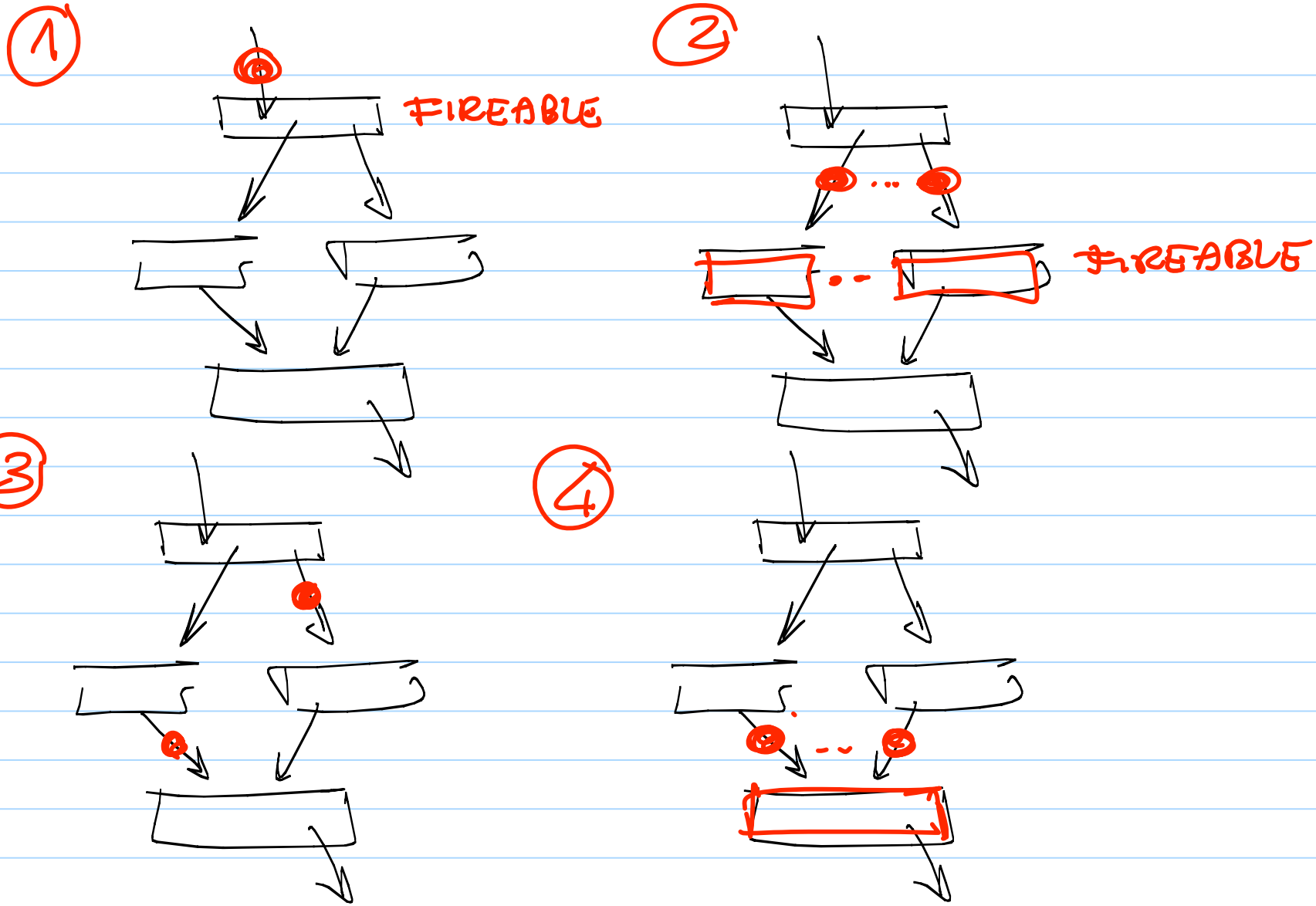
Esempio ??

control flow : `for(i=0; i<N; i++)`

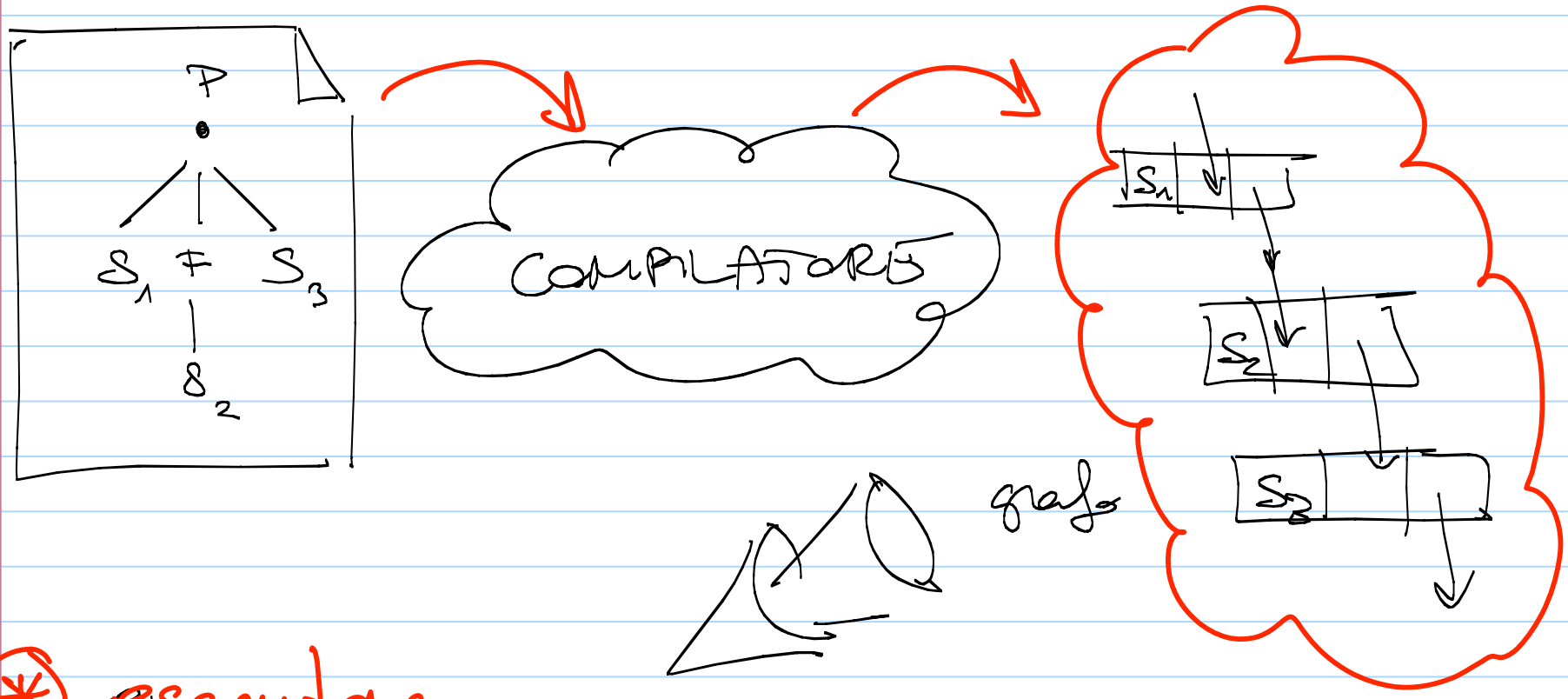
`x[i] = f(x[i]);`

data flow

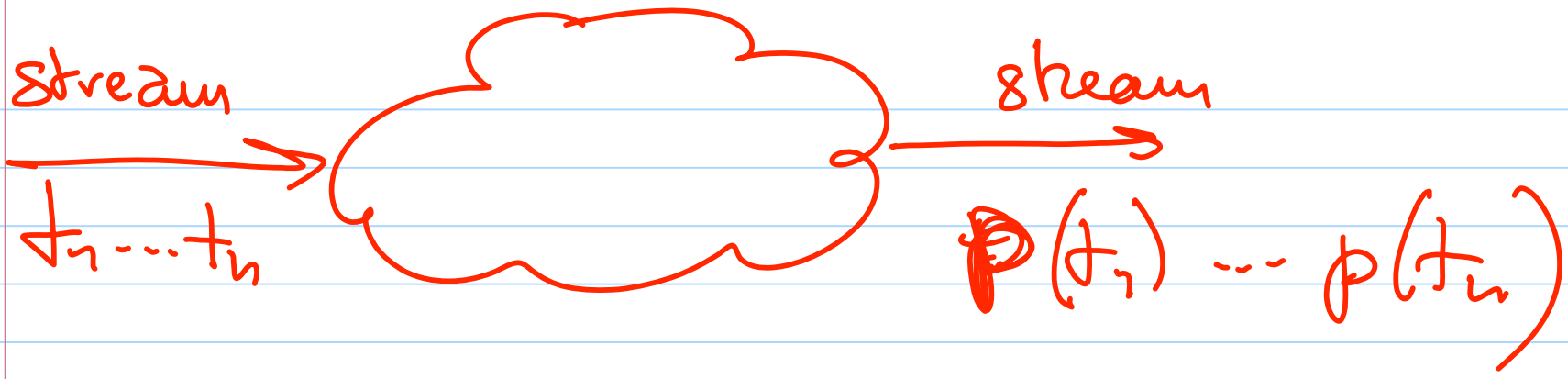



1
FIREABLE
2
FIREABLE
3
4

Come compilo skeleton su macro data flow

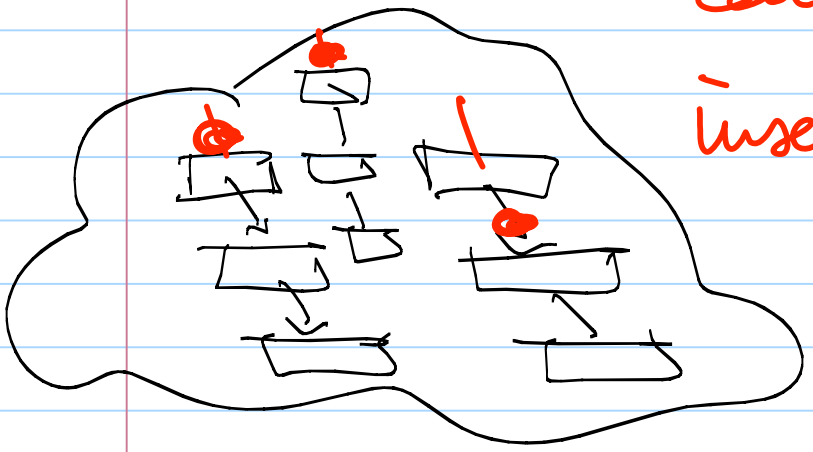


* esecutore parallelo di grafi MDF



$\forall t_i$ istanzio 1 nuova copia del grafo HDF con t_i come token in ingresso

inserisco nel repository delle istruzioni HDF



cerco 1 istruzione fireable e la eseguo